REMARKS

Claims 1 – 17 are pending in the application. Claims 1 – 17 stand rejected. By this amendment, claims 10-12 have been amended.

Claims 10-12 were rejected under 35 U.S.C. 101 as being directed to non-statutory subject matter.

Claims 1-17 were rejected under 35 U.S.C. 102(b) as being anticipated by Sanderson (U.S. Publication No. 2002/0101448).

Applicant now turns to the rejection of claims 10-12 under 35 U.S.C. 101 as being directed toward non-statutory subject matter. Specifically, the Examiner stated that the claims "are directed toward a software tool where the inputs are numbers and the results are also numbers, and/or are directed to a computer program stored in a computer readable medium for implementing the method."

Applicant respectfully disagrees, however, as amended, claims 10-12 include language directed to a tangible output. Specifically, claim 10, for example, now states that the computer executable administrative computing tool application is electronically stored in electronic memory of a computer, and that the "computer is operable to generate and display a graphical user interface utilizing said framework and repository tools in a test environment." Therefore, Applicant respectfully submits that the 35 U.S.C. 101 rejections on claims 10-12 have been traversed.

Applicant now turns to the rejection of claims 1-17 under 35 U.S.C. 102(b) as being anticipated by Sanderson. Claims 1-17 include independent claims 1, 4, 7, 13 and 16. Sanderson teaches a declarative UI generator which receives configuration and workflow data,

and uses this data as context for generation a UI. The Sanderson reference takes a high level approach, describing what tasks are to be performed and not how each task should be performed. The UI generator uses workflow data to describe tasks to be performed, allowing widgets to be selected and arranged. The UI generator also uses configuration data to configure messages sent between the content servers and the data sources. Paragraphs [0012] through [0015]; [0026] through [0028]; [0037]; and [0048].

However, Sanderson does not teach integrating frameworks and repositories, where these frameworks and repositories revolve around data types. In Sanderson, various applications cannot utilize such integrated frameworks and repositories to maintain similarity of presentation, user interaction, and functionality.

Specifically, the Examiner states that Sanderson teaches a UI repository (embodied in data factory 210) containing a UI element (embodied in data element 208). However, Sanderson states that specifications 209 (which specifies the structure and semantics of data elements) are passed to data factories 210, and that these data factories then merely use the information in the specifications to instantiate classes to encapsulate date elements. Para. [0048], [0053]. Thus, these data do not themselves contain attributes of the data elements – this is done by the specifications in Sanderson.

However, these specifications also do not read on the UI repository as claimed in the present Application where the UI repository contains a UI element, which defines data element attributes including data types. Sanderson states that "[c]ontent specifications 209 are provided to specify the structure and semantics of the data elements 208." Para. [0048]. The plural form of the term specification is used repeatedly throughout Sanderson when refering to multiple data

types, which necessarily leads to the conclusion that multiple specifications must be passed from server to client depending on the data types being used. This is one of the explicit deficiencies that the present Application is intended to remedy – having to pass various specifications which are written by the programer is inefficient and leads to non-uniformity.

For example, a UI element can have a data type 'Currency'. The type 'Currency' can refer to an application that controls behavior and functionality. The behavior and functionality and appearance and user interaction will be the same whether the application is an Accounts Receivable application or a Billing Application. In Sanderson, however, a programmer would have to write a specification which includes attributes of the data type 'Currency,' which specification would have to be passed from the server to the client at runtime. This specification may be completely different from another specification that had to be written by the same or a different programmer for a different program that used the data type 'Currency.' Thus, the way the data type 'Currency' is handled by the program in Sanderson may (and likely would) vary across applications, and possibly even across screens in the same application. Thus, as Sanderson teaches only the transmitting of multiple specifications depending on what data types are being used, and does not teach a single integrated UI repository which allows for consistency across screens and programs, and which can interact with other frameworks and repositories, Sanderson does not read on the UI repository as claimed in the present Application.

Additionally, the Examiner states that Sanderson teaches a screen repository as claimed in the present application, which includes screen attributes which define the hierarchical navigation tree structure of screens for a GUI and which further defines what screen will be constructed and defines a GUI component of the screen based on a data type, in that Sanderson

teaches a specification 209 as a screen repository and various workflow and configuration data. However, the screen repository in the present Application "can be called by an application for creating an active screen *that is consistent with all other screen formats and efficiently utilizes other frameworks* to generate the graphical presentation." Para. [0021]. As stated above, the specifications 209 in Sanderson are not uniform, and must be programmed by a programmer and transmitted to the application at runtime. This too, is one of the explicit deficiencies that the present Application is intended to remedy – having to pass various specifications which are written by the programer is innefficient and leads to non-uniform display across programs. Thus, as Sanderson teaches only transmitting multiple specification depending on what data types are being used, and does not teach an single integrated screen repository which allows multiple programs to have a consistent appearance and which can interact with other frameworks and repositories, Sanderson does not read on the integrated screen repository as claimed in the present Application.

Additionally, the Examiner states that Sanderson teaches a data binding framework (embodied in content factory 213) operable to bind data to the UI element and the GUI component based on data type as claimed in the present invention. However, content factory 213 is operable only to create content objects for passage by a messenger between the client and server. Thus, *if* the content factory binds data, it does so only between the client and server, and not between other integrated frameworks and repositories (such as a UI element and GUI component as claimed in the present invention) based on data type so as to ensure consistent handling of data and a consistent appearance of the application, as elaborated on above. Specifically, the claims recite a data binding framework operable to bind data to the UI element

and the GUI component based on data type.

Further, the Examiner states that Sanderson teaches a navigation framework (view 201b, model 201a) controlling generating and displaying of the screens within an application and further builds a navigation tree structure based on the screen attributes. However, Sanderson specifically states "[d]ifferent view components can present the core data of the model component in different ways." Para. [0043]. This is the diametric opposite of what the present Application, as claimed, is designed perform. Specifically the claims recite a navigation framework which controls generating and displaying screens within an application, and which builds a navigation tree structure based on screen attributes. Indeed, the present Application states that "[t]he Navigation Framework provides efficient and consistent navigation through an application." Para. [0020]. Thus, Sanderson does not teach a navigation framework which allows the navigation through various programs to remain consistent and predictable.

Each of independent claims 1, 4, 7, 13, and 16 recite a UI repository containing a UI element defining a data type, a screen repository including screen attributes which define the heirarchical navigation tree structure for a GUI, a data binding framework operable to bind data to the UI element and the GUI component, a GUI framework operable to control how data is handled and processed within the GUI component of the GUI application, and a navigation framework which controls generating and displaying screens within an application, and which builds a navigation tree structure based on screen attributes, all of which are configured or operate based on the data type. Each of these frameworks are interrelated to produce consistent, predictable programs and consistent, predictable screens within programs. Also, independent claim 10 recites a tool for building the integrated frameworks and displaying the resulting

programs which utilizes the frameworks.

The present application teaches a tool for a client-server environment where integrated repositories, or frameworks, have been defined whereby all UI data elements and fields are conformed to fit within the frameworks so that the screens have a predictable appearance and data handling is predictable. It is this predictability that allows a software developer to quickly develop or modify a UI application in a client-server environment without performing rudimentary coding of software. Whereas, Sanderson teaches a different approach where the UI generator browser application allows the data to define the tasks and widgets. As Sanderson does not teach such integrated frameworks, Applicant respectfully submits that independent claims 1, 4, 7, 10, 13 and 16 are not anticipated by Sanderson and are allowable. Additionally, all dependent claims, which contain all of the limitations of the independent claims, are also respectfully submitted to be free from Sanderson and allowable.

If any issue regarding the allowability of any of the pending claims in the present application could be readily resolved, or if other action could be taken to further advance this application such as an Examiner's amendment, or if the Examiner should have any questions regarding the present amendment, it is respectfully requested that the Examiner please telephone Applicant's undersigned attorney in this regard.

Respectfully submitted,

Date: March 25, 2008

Mark E. Stallion
Reg. No. 46,132
Husch Blackwell Sanders LLP
720 Olive Street, 24th Floor
St. Louis, Missouri 63101
(314) 345-6000
ATTORNEYS FOR APPLICANT